**Discussion regarding the choosing of a problem in such a way that it should or must going to require linked lists**

A [linked list](http://www.java67.com/2017/06/5-difference-between-array-and-linked.html?source=post_page---------------------------) is another common data structure that complements the array data structure. Similar to the array, it is also a linear data structure and

stores elements in a linear fashion.

However, unlike the [array](http://www.java67.com/2014/08/what-is-array-data-structure-in-java.html?source=post_page---------------------------), it doesn’t store them in contiguous locations; instead, they are scattered everywhere in memory, which is connected to each other using nodes.

* Elements are also easily removed from a linked list, whereas removing elements from an array leaves empty spaces that are a waste of computer memory. However, unlike arrays that allow random access to the elements contained within them, a link list only allows sequential access to its elements.
* It follows that linked lists should be used for large lists of data where the total number of items in the list is changing. Arrays, on the other hand, are better suited to small lists, where the maximum number of items that could be on the list is known.

**Jeel:-**

These are two problems which can be solved by Linked list.

1. delete loop in the linked list
2. Remove Duplicates from Sorted List

**Akshat:**

A linked list is a data structure that can hold an arbitrary number of data items, and can easily change size to add or remove items. A linked list, at its simplest, is a pointer to a data node. Each data node is then composed of data (possibly a record with several data values), and a pointer to the next node. At the end of the list, the pointer is set to null.

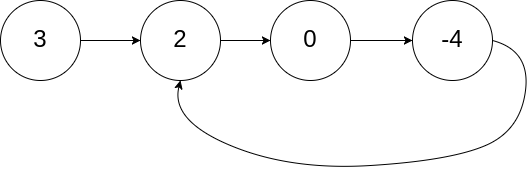
By nature of its design, a linked list is great for storing data when the number of items is either unknown, or subject to change. However, it provides no way to access an arbitrary item from the list, short of starting at the beginning and traversing through every node until you reach the one you want. The same is true if you want to insert a new node at a specific location. It is not difficult to see the problem of inefficiency.

**Anannya:**

1. [Reverse alternate k node in a singly linked list](https://www.geeksforgeeks.org/reverse-alternate-k-nodes-in-a-singly-linked-list/)
2. [Delete last occurrence of an item from linked list](http://quiz.geeksforgeeks.org/delete-last-occurrence-of-an-item-from-linked-list/)
3. [Rotate a linked list.](https://www.geeksforgeeks.org/rotate-a-linked-list/)
4. [Delete n nodes after m nodes of a linked list.](https://www.geeksforgeeks.org/delete-n-nodes-after-m-nodes-of-a-linked-list/)

**Shambhavi**

1. Flattening a linked list
2. .Rotate a linked list
3. How to check if a given linked list contains a cycle. How to find the starting node of the cycle.
4. **Problem Identified:**

Description**:** Given a linked list, determine if it has a cycle in it. To represent a cycle in a given linked list, we use a counter integer pos which represents the position (0-indexed) in the linked list where tail connects to. If pos is -1, then there is no cycle in the linked list. For example: Input: head = [3,2,0,-4], pos = 1 -> Output: true Explanation**:** There is a cycle in the linked list, where tail connects to the second node.

Another example**:** Input: head = [1], pos = -1 -> Output: false

Explanation: There is no cycle in the linked list.

****